IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

ATTY.'S DOCKET: AKKAR=1

In re Application of:) Art Unit: 2137
)
Mehdi-Laurent AKKAR) Examiner: Z. A. Davis
)
Appln. No.: 09/771,967) Washington, D.C.
)
Filed: January 30, 2001) Confirmation No. 2638
)
For: METHOD OF EXECUTING A)
CRYPTOGRAPHIC PROTOCOL)
BETWEEN TWO ELECTRONIC…)

DECLARATION OF INVENTORS UNDER 35 U.S.C. § 1.131

Each of the undersigned, Mehdi-Laurent Akkar and Paul Dischamp, is a co-inventor of the above-identified application and we are collectively the inventors of the above-identified application.

. We understand that the examiner has applied U.S. Patent No. 6,594,761 to Chow in a rejection of the above-identified patent application.

We hereby declare that the aforementioned patent by Chow is not prior art to our invention, inasmuch as we had actually reduced to practice, and thus made our invention, prior to the June 9, 1999 filing date of Chow.

1. In evidence of such reduction to practice, we attach herewith a copy of a description of the invention and a listing of computer code as Exhibit A, having a date (redacted) which is prior to the June 9, 1999, filing date of Chow.

5. The first page of Exhibit A states as follows:

Anti-DPA Improvements in S-BOXes:

Authors: Mehdi-Laurent AKKAR
Paul DISCHAMP

Date: REDACTED

1 - Explanations

- The 8 S-BOXes are processed randomly, so as to:
  - divide the height of peaks by 8 on the signal;
  - avoid a 1-round attack since it is impossible to know which S-BOX is processed.

- Bitwise inverted DES is carried out randomly (one of the characteristics of DES is that this is possible (see Schneier or Stinson)). For that purpose, a second set of bitwise complemented S-BOXes is used both on input and output, so that any attempt to predict which bits circulate within the component will be erroneous. However, at the final XOR output of each round, the output is once again the appropriate one and has to be re-complemented (in the case of an inverted round). If this is done, at some point, whatever the round (whether it is inverted or not), the message will be available in its "clear" form, so that DPA can then be applied. Therefore, before and after each round, the left part of the message is randomly complemented or not (*in the normal case*: inverse, and then inverse, OR non-inverse, and then non-inverse // *in the inverted case:* inverse, and then non-inverse, OR non-inverse, and then inverse). For this purpose, the following steps are carried out: "XORing" is performed with X, and then with X, when nothing has to be changed, and "XORing" is performed with X and X^{-1} (X's complement), thus yielding the inverse. To make this inconspicuous, X is used in such a way that XORing with X and X^{-1} consumes the same amount of processing (in this case, 104 and 151). X could also be chosen randomly.

- Finally, in order to avoid an attack against a large number of messages in which the random generator's bias could be used, the difference between the normal/inverted DES is checked.

The Code of our DES using these countermeasures is as follows:

6. Exhibit A in its entirety was sent, the day after its creation, by mail to our patent attorney, Mr. J. Barbin, at Cabinet Bonnet-Thiron. A copy of the letter is attached as Exhibit B to this declaration.

7. Exhibit B states as follows:

Mr J. Barbin
Cabinet Bonnet-Thirion

12, avenue de la Grande-Armée
75017 Paris

Re : filing of a Soleau enveloppe (CSP99010)

Dear Sirs

Please file on our behalf the enclosed six pages in a Soleau enveloppe in the name of De La Rue Cartes & Systèmes. Thank you in advance and best regards.

D Pottier

8. All of work done in preparation of Exhibit A was done by us or under the direct supervision of at least one of us, and the computer code shown implements the claimed invention.

9. The work reflected in Exhibit A was conducted in France after January 1, 1996, and prior to June 9, 1999.

We hereby declare that all the statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code and the such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Date: __October 27th, 2009__ _____/ Mehdi-Laurent Akkar/______
Mehdi-Laurent Akkar

Date: __October 27th, 2009__ _______/ Paul Dischamp/_________
Paul Dischamp

Améliorations anti-DPA sur les S-BOX:

Auteurs: Mehdi-Laurent AKKAR
Paul DISCHAMP

Date: REDACTED

1 - Explications

- Les 8 S-BOX sont traitées dans un ordre aléatoire, ce qui permet:
  - de diviser la hauteur des pics par 8 sur le signal.
  - d'éviter une attaque en 1 coup car l'on ne sait pas quelle est la S-BOX traitée.

- De manière aléatoire on effectue le DES de manière inversée bit à bit (une des caractéristique du DES est que c'est possible (cf. Schneier ou Stinson)). Pour cela on utilise un deuxième jeu de S-BOX complémentées bit à bit en entrée et en sortie, ce qui fausse toute prédiction sur les bit circulant dans le composant. Cependant à la sortie du xor final de chaque round: la sortie est à nouveau la bonne et il faut (dans le cas d'un round inversé) la recomplémenter. Si l'on procède ainsi, quel que soit le round (inversé ou non), à un moment le message se retrouve en "clair" et l'on peut alors appliquer un DPA. De ce fait avant et après chaque round on complémente ou non de manière aléatoire la partie gauche du message (*dans le cas normal*: inverse puis inverse, OU non inverse puis non inverse // *dans le cas inversé*: inverse puis non inverse, OU non inverse puis inverse). Pour cela on procède ainsi: on "xore" avec X puis avec X quand on ne veut rien faire et l'on "xore" avec X et X^{-1} (complément de X) ce qui donne l'inverse. Pour que ce ne soit pas visible on utilise X tel que le xor avec X et X^{-1} consomme autant (dans ce cas 104 et 151). On pourrait également utiliser X tiré aléatoirement.

- Enfin afin d'éviter une attaque sur un grand nombre de messages où le biais du générateur aléatoire pourrait être utilisé, on contrôle la différence de DES effectué normal/ inversé.

Le Code de notre DES utilisant ces contre-mesures est:

```
EXTRN  DATA  (keydes,xxx)  ; 7 bytes for the deskey
EXTRN  DATA  (inpdes)      ; 8 bytes for the message
EXTRN  DATA  (buffer)      ; 8 bytes for a buffer
EXTRN  DATA  (_shreg)      ; 1 byte for a counter
EXTRN  DATA  (_lpcnt)      ; 1 byte for a counter
EXTRN        DATA  (DES_pointer)        ; 1 byte for the permutation
EXTRN        DATA  (perm)               ; 8 bytes for the permutation table

;*****************************************************************
;   DES 3 bits randomises avec anti DPA (SP et IP)
; valeur de xor 104/151
;*****************************************************************


decrypt:
                                        CALL   IPPERM
                                        MOV    _lpcnt,#010H
```

EXHIBIT A

```
                XRL A,93H                               ; RDS
                MOV keydes+3,A

                MOV R0,#perm  ; Reset perm half to FF
                MOV A,#0FFH
Reset_p:
                MOV @R0,A
                INC R0
                CJNE R0,#perm+4,Reset_p

                MOV _shreg,#7  ; Select a random value between 0 and 7 in keydes
Rand:
                MOV A,_shreg
                CLR C
                SUBB A,#4
                MOV A,_shreg
                JNC MSN

                ADD A,#keydes
                MOV R0,A
                MOV A,@R0
                SJMP MSN_end
MSN:
                ADD A,#keydes-4
                MOV R0,A
                MOV A,@R0
                SWAP A
MSN_end:
                ANL A,#07H
                ADD A,#perm             ; Position in perm using this random value
                MOV R0,A
Compare:
                MOV A,@R0
                MOV R1,A

                MOV A,keydes+4
                CJNE A,#0FFH,p_2

                CJNE R1,#0FFH,Next_p    ; Check if position already used

                MOV A,_shreg            ; Write value in position
                SWAP A
                ORL A,#0FH
                SJMP Next_index
p_2:
                MOV A,R1                ; Second permutation
                ANL A,#0FH
                CJNE A,#0FH,Next_p

                MOV A,@R0               ; Write value in position
                ANL A,#0F0H
                ORL A,_shreg
                SJMP Next_index
Next_p:
                DEC R0                  ; Move on to next position
                CJNE R0,#perm-1,Compare
                MOV R0,#perm+7
                SJMP Compare
Next_index:
                MOV @R0,A
                DEC _shreg
                MOV A,_shreg
                CJNE A,#0FFH,Rand

                MOV A,keydes+4          ; Loop for second permutation
                CJNE A,#0FFH,p_end
                JMP Create_perm
p_end:
                MOV R0,#perm
                MOV R1,#keydes+4
Retr_prm:
                MOV A,@R1
                ANL A,@R0               ; Retrieve data saved in keydes during first permutation
                MOV @R0,A
                INC R0
                INC R1
                CJNE R0,#perm+3,Retr_prm

                MOV A,DES_pointer
                ANL A,@R0
                MOV @R0,A
                MOV DES_pointer,#0      ; Reset DES_pointer

                MOV R0,#keydes          ; Clear keydes zone
                CLR A
Clear_kd:
                MOV @R0,A
                INC R0
                CJNE R0,#keydes+7,Clear_kd

                MOV R1,#perm+7          ; Bit position
Byte_pos:
                MOV A,@R1
                ANL A,#0F0H
                SWAP A
                MOV _shreg,A
                JZ Try_again

                MOV R2,#perm+7          ; Byte position
Next_move:
                MOV R0,AR2              ; Position on input buffer
                MOV A,@R0
                ANL A,#0FH
                MOV _lpcnt,A

                ADD         A,#buffer
                MOV R0,A
                MOV A,@R0
                MOV R0,A                ; Save byte in R0

                MOV DPTR,#MASK
                MOV A,_shreg
                MOVC A,@A+DPTR
                ANL A,R0
                MOV B,A                 ; B now contains masked bit

                MOV A,_lpcnt            ; Shift bit to final position
                CLR C
                SUBB A,_shreg
                JZ Copy
                JNC Shift_left
```

```
                                RRC A
                                MOV C,BIT14
                                RRC A
                                RR          A
                                RR          A
B0_END:
                                ORL         A,inpdes
                                MOV inpdes,A
                                SJMP PC2_Loop
B0_LSN:
                                CLR         A
                                MOV C,BIT24
                                RLC A
                                MOV C,BIT1
                                RLC A
                                MOV C,BIT5
                                RLC A
                                SJMP B0_END
B1_MSN:
                                CLR A
                                MOV C,BIT15
                                RRC A
                                MOV C,BIT28
                                RRC A
                                MOV C,BIT3
                                RRC A
                                RR          A
                                RR          A
B1_END:
                                ORL A,inpdes+1
                                MOV inpdes+1,A
                                SJMP PC2_Loop
B1_LSN:
                                CLR A
                                MOV C,BIT6
                                RLC A
                                MOV C,BIT21
                                RLC A
                                MOV C,BIT10
                                RLC A
                                SJMP B1_END
B2_MSN:
                                CLR A
                                MOV C,BIT12
                                RRC A
                                MOV C,BIT19
                                RRC A
                                MOV C,BIT23
                                RRC A
                                RR          A
                                RR          A
B2_END:
                                ORL         A,inpdes+2
                                MOV inpdes+2,A
                                SJMP PC2_Loop
B2_LSN:
                                CLR A
                                MOV C,BIT4
                                RLC A
                                MOV C,BIT26
                                RLC A
                                MOV C,BIT8
                                RLC A
                                SJMP B2_END
B3_MSN:
                                CLR A
                                MOV C,BIT27
                                RRC A
                                MOV C,BIT7
                                RRC A
                                MOV C,BIT16
                                RRC A
                                RR          A
                                RR          A
B3_END:
                                ORL         A,inpdes+3
                                MOV inpdes+3,A
                                JMP PC2_Loop
P2_JTab:
                                SJMP B0_MSN
                                SJMP B0_LSN
                                SJMP B1_MSN
                                SJMP B1_LSN
                                SJMP B2_MSN
                                SJMP B2_LSN
                                SJMP B3_MSN
                                SJMP B3_LSN
                                SJMP B4_MSN
                                SJMP B4_LSN
                                SJMP B5_MSN
                                SJMP B5_LSN
                                SJMP B6_MSN
                                SJMP B6_LSN
                                SJMP B7_MSN
                                SJMP B7_LSN
B3_LSN:
                                CLR A
                                MOV C,BIT20
                                RLC A
                                MOV C,BIT13
                                RLC A
                                MOV C,BIT2
                                RLC A
                                SJMP B3_END
B4_MSN:
                                CLR A
                                MOV C,BIT31
                                RRC A
                                MOV C,BIT52
                                RRC A
                                MOV C,BIT41
                                RRC A
                                RR          A
                                RR          A
B4_END:
                                ORL         A,inpdes+4
                                MOV inpdes+4,A
                                JMP PC2_Loop
B4_LSN:
                                CLR A
                                MOV C,BIT37
```

```
;           SP SUBROUTINE
;
;           Input  INPDES 0---7
;           Output BUFFER 0---3
;
;*****************************************************************

                                        mov         A,R3
                                        xrl         buffer+0,A
                                        xrl         buffer+1,A
                                        xrl         buffer+2,A
                                        xrl         buffer+3,A

                                        mov         A,R4
                                        xrl         buffer+0,A
                                        xrl         buffer+1,A
                                        xrl         buffer+2,A
                                        xrl         buffer+3,A


splop1:                                 mov         R6,#8
                                        mov         A,093H
                                        XRL         A,094H
                                        anl         A,#00000111b                         ; CM  rdm ordre Sbox (anl A,#00000111b)
                                        mov         B,#10
                                        mul         AB
                                        mov         DPTR,#D2
                                        JMP         @A+DPTR

MLA2211:                  LJMP   SWAPLR

D2:                                     mov         R0,#0
                                        mov         R1,#inpdes
                                        lcall       SPLOP
                                        mov         A,R6
                                        JZ          MLA2211

                                        mov         R0,#4
                                        mov         R1,#inpdes+1
                                        lcall       SPLOP
                                        mov         A,R6
                                        JZ          MLA2211

                                        mov         R0,#8
                                        mov         R1,#inpdes+2
                                        lcall       SPLOP
                                        mov         A,R6
                                        JZ          MLA2211

                                        mov         R0,#12
                                        mov         R1,#inpdes+3
                                        lcall       SPLOP
                                        mov         A,R6
                                        JZ          MLA2211

                                        mov         R0,#16
                                        mov         R1,#inpdes+4
                                        lcall       SPLOP
                                        mov         A,R6
                                        JZ          MLA2211

                                        mov         R0,#20
                                        mov         R1,#inpdes+5
                                        lcall       SPLOP
                                        mov         A,R6
                                        JZ          MLA2211

                                        mov         R0,#24
                                        mov         R1,#inpdes+6
                                        lcall       SPLOP
                                        mov         A,R6
                                        JZ          MLA2211

                                        mov         R0,#28
                                        mov         R1,#inpdes+7
                                        lcall       SPLOP
                                        mov         A,R6
                                        JZ          MLA221

                                        mov         R0,#0
                                        mov         R1,#inpdes
                                        lcall       SPLOP
                                        mov         A,R6
                                        JZ          MLA221

                                        mov         R0,#4
                                        mov         R1,#inpdes+1
                                        lcall       SPLOP
                                        mov         A,R6
                                        JZ          MLA221

                                        mov         R0,#8
                                        mov         R1,#inpdes+2
                                        lcall       SPLOP
                                        mov         A,R6
                                        JZ          MLA221

                                        mov         R0,#12
                                        mov         R1,#inpdes+3
                                        lcall       SPLOP
                                        mov         A,R6
                                        JZ          MLA221

                                        mov         R0,#16
                                        mov         R1,#inpdes+4
                                        lcall       SPLOP
                                        mov         A,R6
                                        JZ          MLA221

                                        mov         R0,#20
                                        mov         R1,#inpdes+5
                                        lcall       SPLOP
                                        mov         A,R6
                                        JZ          MLA221
```

DeLaRue

```
;**************************************************************

IPPERM:


                                  MOV    R0,#inpdes+7·

IP2:
                                  MOV    R1,#buffer
                                  MOV    A,@R0

IP1:
                                  RLC    A
                                  RLC    A
                                  XCH    A,@R1
                                  RLC    A
                                  XCH    A,@R1
                                  INC    R1
                                  CJNE   R1,#buffer+8,IP1
                                  DEC    R0
                                  CJNE   R0,#inpdes-1,IP2

; Test inversion



                                  mov          A,093H
                                  xrl          A,094H
                                  mov          R3,A
                                  anl          A,#1
                                  mov          B,#3
                                  mul          AB
                                  inc          A


iiinv:

                                  mov          DPTR,#MLA1
MLA1:
                                  JMP          @A+DPTR
                                  JMP          norma
                                  JMP          inver

norma:
                                  mov          A,XXX+1
                                  inc          A
                                  CJNE         A,#125,norma2
;                                 JMP          inver                                 ; virer le pt virg
norma2:                 mov       XXX+1,A

                                  mov          R4,#104
                                  mov          R5,#104                               ; CM normalt 104 et SPTAB
                                  mov          DPTR,#SPTAB0
                                  mov          xxx,DPH

                                  jmp          MLA11


inver:                  mov       A,XXX+1
                                  dec          A
                                  CJNE         A,#115,inver2
;                                 JMP          norma                                      ; Virer le pt virg
inver2:                 mov       XXX+1,A

                                  mov          R4,#104
                                  mov          R5,#151                               ; CM normalt 151 et IPTAB
                                  mov          DPTR,#IPTAB0
                                  mov          xxx,DPH

                                  jmp          MLA11·


MLA11:
                                  mov          A,R3
                                  rr           A
                                  anl A,#1                                  ; CM Attaque Rnd 16 sur l'inverse   anl A,#1
                                  mov          B,#3
                                  mul          AB
                                  inc          A

                                  mov          DPTR,#MLA6662
MLA6662:       JMP      @A+DPTR
                                  JMP          norma666
                                  JMP          inver666

norma666:  mov R3,#104
                                  JMP          MLA666

inver666:      mov      R3,#151
                                  JMP          MLA666

MLA666:

                                  call inverse

; Fin test Inversion


                                  RET
;**************************************************************
;          IP-1 SUBROUTINE
;
;          Input  BUFFER 0---7
;          Output INPDES 0---7
;
;  Permutation table:    40  8 48 16 56 24 64 32
;                        39  7 47 15 55 23 63 31
;                        38  6 46 14 54 22 62 30
;                        37  5 45 13 53 21 61 29
;                        36  4 44 12 52 20 60 28
;                        35  3 43 11 51 19 59 27
;                        34  2 42 10 50 18 58 26
;                        33  1 41  9 49 17 57 25
;**************************************************************

IPMIN1:
                                  CALL         MOVE_PERM
```

```
;
RIGHSH:
                                        MOV R0,#keydes
                                        MOV A,keydes+3
                                        RL A
                                        RL A
                                        RL A
                                        MOV B,A
RSHIF:
                                        MOV A,@R0
                                        RR A
                                        MOV R1,A
                                        ANL A,#07FH
                                        XRL A,B
                                        ANL B,#07FH
                                        XRL A,B
                                        MOV @R0,A
                                        MOV B,R1
                                        INC R0
                                        CJNE R0,#keydes+7,RSHIF

                                        MOV A,B
                                        RL A
                                        RL A
                                        RL A
                                        RL A
                                        XCH A,keydes+3
                                        ANL A,#0F7H
                                        XRL A,keydes+3
                                        XCH A,keydes+3
                                        ANL A,#0F7H
                                        XRL A,keydes+3
                                        MOV keydes+3,A

                                        DJNZ    R2,RIGHSH
                                        RET

;**********************************************************************
;
;       COMPUTE NUMBER OF SHIFT
;
; ROUND      : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
;
; ENCRYPTION: 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 1
;
; DECRYPTION: 1 2 2 2 2 2 2 1 2 2 2 2 2 2 1 1
;
;**********************************************************************

NBSHIFT:
                                        MOV    A,_shreg
                                        RL     A
                                        MOV    C,ACC.0
                                        MOV    _shreg,A
                                        XRL    A,#3FH
                                        JNZ    SHNE
                                        MOV    _shreg,#7EH
SHNE:
                                        MOV    A,_lpcnt
                                        DEC    A
                                        DEC    A
                                        JNZ    SHNE1
                                        MOV    _shreg,#7EH
SHNE1:
                                        CLR    A
                                        INC    A
                                        JNC    SHNE2
                                        INC    A
SHNE2:
                                        MOV    R2,A
                                        RET

adddptr:
                                        mov                     A,DPL
                                        add                     A,#64
                                        mov                     DPL,A
                                        JNC                     finadddptr
                                        inc                     DPH
finadddptr: ret

;**********************************************************************
;*          SP TABLES
;**********************************************************************

;_TABLES    SEGMENT CODE
;  RSEG    _TABLES

CSEG AT 08D00H

SPTAB0:       DB     0D8H,0D7H,083H,03DH,01CH,08AH,0F0H,0CFH
                         DB     072H,04CH,04DH,0F2H,0EDH,033H,016H,0E0H
                         DB     08FH,028H,07CH,082H,062H,037H,0AFH,059H
                         DB     0B7H,0E0H,000H,03FH,009H,04DH,0F3H,094H
                         DB     016H,0A5H,058H,083H,0F2H,04FH,067H,030H
                         DB     049H,072H,0BFH,0CDH,0BEH,098H,081H,07FH
                         DB     0A5H,0DAH,0A7H,07FH,089H,0C8H,078H,0A7H
                         DB     08CH,005H,072H,084H,052H,072H,04DH,038H

SPTAB1:       DB     0D8H,035H,006H,0ABH,0ECH,040H,079H,034H
                         DB     017H,0FEH,0EAH,047H,0A3H,08FH,0D5H,048H
                         DB     00AH,0BCH,0D5H,040H,023H,0D7H,09FH,0BBH
                         DB     07CH,081H,0A1H,07AH,014H,069H,06AH,096H
                         DB     047H,0DAH,07BH,0E8H,0A1H,08FH,098H,046H
                         DB     0B8H,041H,045H,09EH,05EH,020H,0B2H,035H
                         DB     0E4H,02FH,09AH,0B5H,0DEH,001H,065H,0F8H
                         DB     00FH,0B2H,0D2H,045H,021H,04EH,02DH,0DBH

SPTAB2:       DB     0DBH,059H,0F4H,0EAH,095H,08EH,025H,0D5H
                         DB     026H,0F2H,0DAH,01AH,04BH,0A8H,008H,025H
                         DB     046H,016H,06BH,0BFH,0ABH,0E0H,0D4H,01BH
                         DB     089H,005H,034H,0E5H,074H,07BH,0BBH,044H
                         DB     0A9H,0C6H,018H,0BDH,0E6H,001H,069H,05AH
                         DB     099H,0E0H,087H,061H,056H,035H,076H,08EH
                         DB     0F7H,0E8H,084H,013H,004H,07BH,09BH,0A6H
                         DB     07AH,01FH,06BH,05CH,0A9H,086H,054H,0F9H
```

DeLaRue

DeLaRue

Paris, REDACTED

De:

D. POTTIER
tel. 33 1 49 69 24 66
fax 33 1 49 69 25 03

DE LA RUE CARD SYSTEMS

3-5, avenue Galliéni
94250 GENTILLY - France

Telephone : + 33 (0)1 53 62 51 00
Marketing fax : + 33 (0)1 49 69 25 02
R&D fax : + 33 (0)1 49 69 25 03
http://www.delarue.com

à:

Monsieur J. BARBIN
Cabinet Bonnet-Thirion
12 avenue de la Grande Armée
75017 PARIS

Votre Ref.	—
Notre Ref.	DLRCS/DP/DEV/dp/99108

Objet: Dépôt d'une enveloppe Soleau (CSP 99010)

Monsieur,

Je vous prie de bien vouloir déposer pour nous les **six feuilles** jointes dans une enveloppe Soleau au nom de De La Rue Cartes & Systèmes.

Vous en remerciant d'avance, je vous prie de croire, Monsieur, à l'assurance de mes sentiments distingués.

D. Pottier

EXHIBIT B